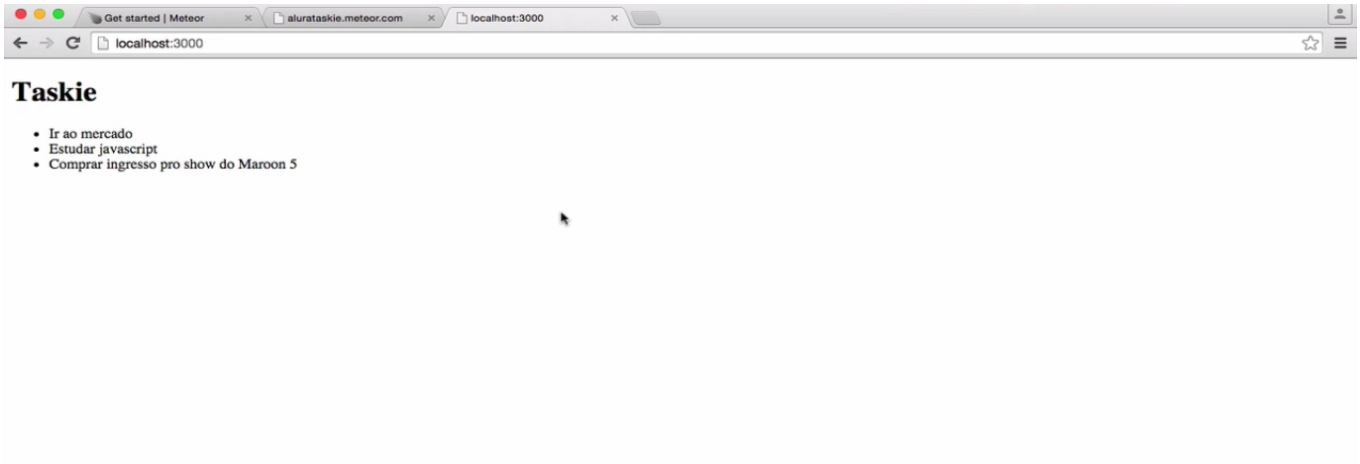
02

Exibindo tarefas e o MongoDB

Nosso próximo passo é não deixar mais os dados que aparecem na tela da internet fixos. Eles, normalmente, devem vir de um banco de dados.



Fazer um banco de dados no *Meteor* é muito simples, ele já vêm com um *Mongo DB* plugado. Basta criar umas três linhas de código para isso funcionar.

Vamos voltar na *Sublime*. A primeira coisa é criar um banco de dados para guardar as tarefas.

Criaremos uma nova pasta, "models", que guardaremos conteúdos tanto do servidor quanto do cliente. O *Meteor* nos permite executar código do cliente e do servidor ao mesmo tempo.

Criaremos um arquivo chamado "tarefas.js". Nesse código escreveremos

```
Tarefas = new Mongo.Collection("tarefas")
```

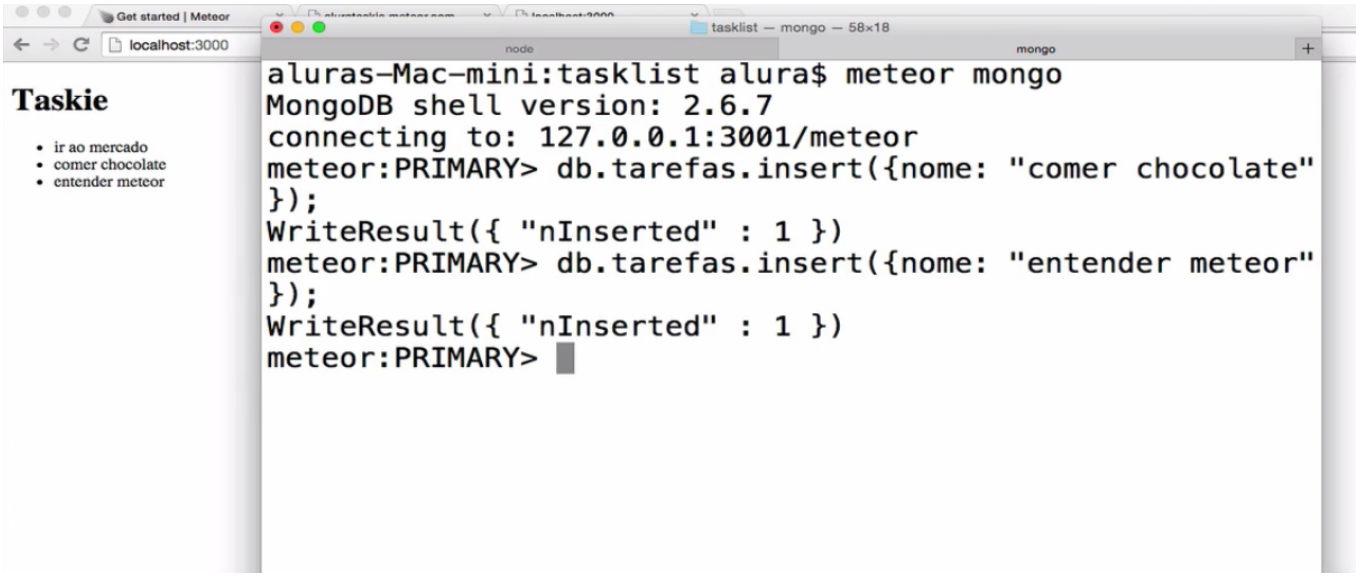
Em uma linha de código o *Meteor* cria a coleção "Tarefas" no *Mongo.DB*.

Agora, podemos ir na lista, na aba `lista.js` e excluir a sequência dos itens que tínhamos posto na nossa lista `{nome: "Ir ao mercado"}` , `{nome:"Comprar sorvete"}` e `{nome: "estudar"}` . E acrescentaremos o objeto `Tarefas` , escrevendo da mesma maneira que escrevemos na aba `tarefas.js` , pois, ele é um objeto que agora está visível para todos. Vamos colocar um `find` e passaremos um mapa com as condições, os "where", só que ele estará vazio por enquanto. Teremos, `return Taferas.find({})` .

Se formos ver a nossa página da internet veremos que ela estará vazia. E isso já era esperado devido ao nosso banco estar sem nada.

Podemos ir no terminal e inserir algumas coisas novas. Para isso criaremos um novo terminal. Inserimos `meteor mongo` e ele dará um console, onde poderemos digitar coisas diretamente para o *Mongo*.

Vamos inserir na linha do `meteor:Primary` um `db.tarefas` . Cuidando para que o nome coincida com o nome da lista no *mongo*. Vamos inserir também a tarefa, no caso ir ao mercado. Teremos, `db.tarefas.insert({nome: "comer chocolate"})` . Vai aparecer um `WriteResult` .



A medida que formos inserindo as novas tarefas aqui na nossa tela poderemos reparar que o *browser* não dá *refrash* na tela. Para observar isso, basta colocar a tela do *Meteor* acima do browser e inserir um novo objeto para reparar.

O nome disso é programação reativa.

Quando digitamos no *Mongo*, na aba "lista.js", um `Tarefas.find` ele pega todas as tarefas e fica esperto para saber se teve mudanças no banco. Isto é, fica o tempo todo sincronizando. O servidor do *Meteor avisa* o cliente e o cliente redesenha a listagem e isso ocorre todo o tempo. Isso é programação reativa.